Introduction

In our efforts to address the Big Data challenges of climate science, we are moving toward a notion of Climate Analytics-as-a-Service (CAaaS) and delivery of capabilities through the NASA Climate Data Services Application Programming Interface (CDS API). The CDS API provides a uniform semantic treatment of the combined functionalities of large-scale data management, data-proximal analytics, and related services. The CDS API combines concepts from the Open Archive Information Systems (OAIS) reference model, object-oriented programming APIs, and Web 2.0 resource-oriented APIs.

OAIS is sponsored by the Consultative Committee for Space Data Systems (CCSDS): http://public.ccsds.org/publications/archive/650x0m2.pdf

Download-Installation

CDS services are available by accessing the CDS API. The CDS API can be executed using either direct web service calls, a Python script of sequential commands, or a full featured Python application. The CDS client package contains everything needed to make remote web services calls.

```
Source Code Download Page
The MERRA/AS client stack and API is released under the following disclaimer:

NASA's Goddard Space Flight Center
NASA/GSFC/CISTO
8800 Greenbelt Rd, Greenbelt, MD 20771

THIS SOFTWARE AND ITS DOCUMENTATION ARE CONSIDERED TO BE IN THE PUBLIC DOMAIN AND THUS ARE AVAILABLE FOR UNRESTRICTED PUBLIC USE.
THEY ARE FURNISHED "AS IS." THE AUTHORS, THE UNITED STATES GOVERNMENT, ITS INSTRUMENTALITIES, OFFICERS, EMPLOYEES, AND AGENTS MAKE
EXPRESS OR IMPLIED, AS TO THE USEFULNESS OF THE SOFTWARE AND DOCUMENTATION FOR ANY PURPOSE. THEY ASSUME NO RESPONSIBILITY (1) FOR
USE OF THE SOFTWARE AND DOCUMENTATION; OR (2) TO PROVIDE TECHNICAL SUPPORT TO USERS.
```

Download Current Stable Release: cds_client_0.8.3.tar

Decompress the tar file (e.g., tar -xvf cds_client_*version*.tar) and see the enclosed 'usage.txt' for detailed usage.

Requirements

- Internet access
- Python 2.6+ (only required if accessing the CDS API via Python scripts or applications)
- Internet browser (only required if accessing the CDS API via direct web service URLs)

Climate Data Services - Application Programming Interface

The Climate Data Services (CDS) Application Programming Interface (API) defines the set of actions (e.g., methods) that can be requested.

CDS-API-Method-Overview:

Each CDS API method, which corresponds to a specific web service URL context, is described below.

1. Order – Request data from a pre-determined service request (asynchronous).
2. Status – Track progress of service activity.
3. Download – Retrieve a Dissemination Information Package (DIP).
4. Ingest – Submit/register a Submission Information Package (SIP).
5. Execute – Initiate a service-definable extension. Allows for parameterized growth without API change.
6. Query – Retrieve data from a pre-determined service request (synchronous).

Climate Data Services Library

The CDS Library is a Python class (i.e., CDSApi) that prepares web service calls while abstracting the details. This class, which can be imported into any Python application, includes basic utilities and extended utilities. Basic utilities call a single CDS API method (e.g., order) and can also be included in Python scripts and applications. Extended utilities may weave together multiple CDS API methods (e.g., avg) and are intended for use as convenience functions in Python scripts and applications. For sample invocations, see the MERRA/AS method overview.

CDS Basic Utilities:

Basic utilities are designed for the CDS API Web Service calls, or Python scripts and applications.

Order

Request data from a pre-determined service request (asynchronous).

Typically initiates an asynchronous Map Reduce job that generates an aggregate NetCDF file according to the user-specified arguments.

Arguments:

```
service: The name of the target service.  For example:  MAS
request:  The service request to be initiated (e.g., GetVariableBy_TimeRange_SpatialExtent_VerticalExtent)
parameters:  request-specific parameter list
```

Returns:

```
sessionId: session id. A unique UUID that is used to retrieve session status (e.g., f654c35c-f223-4787-a947-8787f532d3fe)
sessionStatus:  One word status of the session (e.g., Received, Completed, Failed)
sessionStatusDetail:  Description of the session (e.g., Order failed due to lack of privileges)
```

Order-Services:

Service	Description
MAS	MERRA/AS service, used to order and download subsetted MERRA/AS data

Status

Track progress of service activity.

Status is a method for session tracking. Once initiated, a session enters into different processing stages. The Status method offers the client a way of querying the current state of the session.

Arguments:

```
service: The name of the target service.  For example:  MAS
sessionId:  Session ID
```

Returns:

```
sessionId:  Session id
sessionStatus:  One word status of the session (e.g., Processing, Completed, Failed)
sessionStatusDetail:  Description of the session (e.g., MapReduce job in progress)
```

Status-Values:

Status	Description
Received	Order successfully received
Processing	Order in progress
Completed	Order/ingest was successfully completed
Failed	Order/ingest failed

Download

Retrieve a Dissemination Information Package (DIP).

The Download method provides a means for retrieving packages associated with a sessionId from a service provider. In a typical asynchronous transaction, such as those associated with Order, this method obtains results from a service request.

Arguments:

```
service: The name of the target service.  For example:  MAS
sessionId:  Session ID for the order
DIP: Optional dissemination information package (name!type!content).
    When this parameter is empty, all packages associated with the supplied sessionId
    are returned.  When a package name is specified within the DIP parameter, the
    service provider returns only the associated package.
```

Returns:

```
DIP:  Requested dissemination information package
```

Download-Services:

Service	Description
MAS	MERRA/AS service, used to order and download subsetted MERRA/AS data
PS	Persistence service, used to ingest, search for, and download a submission information package (SIP)

Ingest

Submit/register a Submission Information Package (SIP).

Provide a SIP to be persisted by the service. A SIP will typically be a script or application to be shared, but any ingested artifact will be stored.

Arguments:

```
service: The name of the target service.  For example:  MAS
```

```
operation: operation to perform (e.g., Put)
SIP: Submission information package (name!type!content)
sessionId:  Optional session ID for the SIP if the operation is a result of a previous transaction (e.g., ingest parts of a s
overwrite:  If 'true' overwrite previous copy
```

Returns:

```
sessionId:  Session id
sessionStatus:  One word status of the session (e.g., Received, Completed, Failed)
sessionStatusDetail:  Description of the session (e.g., Ingest failed due to insufficient HDFS storage space)
```

Ingest-Services:

Service	Description
PS	Persistence service, used to ingest, search for, and download a submission information package (SIP)

Execute

Initiate a service-definable extension. Allows for parameterized growth without API change.

The Execute method implementation is optional for service providers. It is a service extension mechanism that allows a service provider to offer additional services that are accessed in a common way.

Arguments:

```
service: The name of the target service.  For example:  MAS
methodName: The name of the method to be processed.  For example, ApplyDataRetentionPolicyToData (methods supported by iRODS
parameters:  1 or more parameters of the format (name!value).
```

Returns:

```
sessionId: session id.
sessionStatus:  One word status of the session (e.g., Received, Completed, Failed)
resultSet: execute results in XML format
```

Execute-Services:

Service	Description
PS	Persistence service, used to ingest, search for, and download a submission information package (SIP)

Query

Retrieve data from a pre-determined service request (synchronous).

The Query method executes a predefined service request that returns data in an XML instance package that conforms to a predefined standard schema

Arguments:

```
service: The name of the target service.  For example:  MAS
request: The name of the query to be processed.  For example, GetAvailableOperatorsByParameter  (queries supported by MAS Ser
parameters:  1 or more parameters of the format (name!value).
```

Returns (resultSet or fault)

```
resultSet: query results in XML format    OR
fault:  Error code (e.g., E_InvalidParameter)
```

Query-Services:

Service	Description
PS	Persistence service, used to ingest, search for, and download a submission information package (SIP)